数据中心全系统模拟方法研究

胡农达 付斌章 隋秀峰 李龙 朱晓东 李涛 陈明宇 张立新

摘要 随着云计算的发展，数据中心快速崛起并给设计者和管理者带来了很多新的挑战。模拟作为一种研究计算机系统的有效方法已被用于数据中心的研究，但是现有的模拟器不能对整个数据中心进行模拟。从头开发一个完整数据中心模拟器复杂性大、开发周期长，并且其可信度和有效性还需要经过长时间的考验。因此，本文探索重用现有模拟器进行数据中心全系统模拟的思路，提出了一种数据中心全系统的模拟框架，它可以通过整合现有的模拟器来构建完整的数据中心模拟器。

关键词 数据中心 模拟器 同步 通信 并行

1 介绍

云计算，也就是基础架构即服务、平台即服务和软件即服务，正在变革和重塑着今天的 IT 产业，并影响着 IT 基础设施的设计[1]。由于云计算具有灵活、可扩展和低成本的特点，很多传统 IT 服务开始向其上迁移。这促进了对云计算的物理基础设施——数据中心的需求极大增长，世界各地出现了很多能容纳数万甚至数十万服务器的大型数据中心。但是大型数据中心的构建、运作和管理也给设计者和管理者带来了很多挑战[2]。这包括：可扩展架构的设计、虚拟化、系统配置、故障侦测和诊断、功耗和热能管理和安全问题等等。工业界和学术界都投入了大量的精力对这些问题进行研究。

对于数据中心的研究，主要有两种手段：测试和模拟。测试一般采用测试床进行实验。测试床是微缩的数据中心，可以再现数据中心的很多真实负载和场景，以反映实际的需求与问题。但是由于其规模小，很难对大规模复杂场景进行刻画，尤其是难以进行可扩展性方面的实验。而且，构建测试床需要大量的时间、人力和经济上的投入，很多研究者不具备这样的条件。更进一步的问题是，测试床的很多部分对研究者来说是黑盒，要观察其运行细节或采集性能数据与统计信息是十分困难的。模拟的方法，正好弥补了测试床方法的这些缺点。首先，模拟环境往往只需要一台或数台主机，在几小时到几天的时间内就可以搭建完成；其次，模拟的系统规模很容易扩展到数千甚至上万节点；再次，被模拟的系统对研究者来说是完全可见的，很容易从中收集性能数据和统计信息。除此以外，研究者很容易对模拟的系统进行修改，加入新的设计，以验证自己新的想法。模拟方法由于有这些优点，在数据中心的研究中被广泛使用。

在对计算机系统的研究过程中，已经开发了大量的模拟器。根据这些模拟器模拟的功能可以将他们分为局部模拟器和系统模拟器两个类别。局部模拟器对计算机系统的某个部分进行细致模拟，比如处理器模拟器 M-Sim[3]，缓存模拟器 Dinero IV[4]，存储系统模拟器 Ruby[5]，磁盘模拟器 DiskSim[6]，网络模拟器 NS3[7]，能耗模拟器 Wattch[8]等。系统模拟器对整个计算机系统进行模拟，模拟的部件有处理器、缓存、内存、磁盘、控制器和 I/O 总线等。常见的系统模拟器有 Simics[9]、SimpleScalar[10]、SimOS[11]、M5[12]、GEM5[13]等。大多数系统模拟器仅对单节点进行模拟。此外，随着云计算的研究变热，还出现了一些针对云服务和数据中心的模拟器，比如 MDCSim[14]、CloudSim[15]、GreenCloud[16]、NetworkCloudSim[17]等。

尽管大量模拟器已经被开发并在计算机系统的研究中广泛使用，但是大多数模拟器仅对

系统的某一部分或系统的某一方面功能进行单独模拟。即使是系统模拟器，大部分也只能对单个节点或少数互连的节点进行模拟。而在数据中心中，大量的高通量应用，比如 MapReduce 应用，往往同时运行在大量节点上，其性能由节点、网络以及它们之间的交互共同决定。此外，数据中心的管理，比如负载均衡、容灾备份等，也往往需要考虑数据中心各部分的信息。以负载平衡时将虚拟机从一个物理节点迁移到另一个物理节点的情形为例：如果仅考虑处理器利用率的信息，调度结果很可能是将虚拟机迁移到一个处理器资源富裕但只有少量网络带宽可用的物理节点，这显然会对系统运行的效率产生不利影响。由此可见，如果有一个模拟环境能对整个数据中心进行模拟，将极大地促进数据中心的相关研究。但是从头开发一个针对整个数据中心的模拟器，复杂度大，开发周期长，并且后续的验证和调优代价也很高。然而，如果能够重用那些久经考验的模拟器，将他们整合成一个数据中心的模拟环境，有可能避免这些问题。

因此，我们提出了一种全系统的模拟框架，通过该框架我们可以整合现有的诸如网络模拟器、存储模拟器、处理器模拟器等各类局部模拟器以及一些单节点系统模拟器来构建自己的数据中心模拟器。我们的基本思想和方法是在原有的各独立运行的模拟器的下方，提供一层多模拟器整合机制，将他们粘合成一个完整的单一的数据中心模拟器。通过该方法构建的数据中心全系统模拟器具有以下特点：

– 全系统模拟：该模拟器支持对 CPU，高速缓存、内存系统、I/O 子系统、存储子系统以及数据中心网络的模拟，并且可以运行不加修改的操作系统、用户应用程序以及网络协议；

– 分布式模拟：为了加快仿真速度和扩大仿真规模，该模拟器可以支持基于消息通信的分布式并行仿真，并且保证整个系统的同步；

– 事件驱动：采用事件驱动方式可以加速仿真。

本文第 2 节介绍模拟器的相关工作；第 3 节介绍我们提出的数据中心的全系统模拟框架的设计；第 4 节给出了一个实现及其实验；最后是总结。

2 相关工作

2.1 计算机系统模拟器

已有的计算机系统模拟器中，以下三款有一定的代表性。

– **GEM5**

GEM5[13]是一款模块化的离散事件驱动的计算机系统仿真平台，其开发语言为 C++和 python。GEM5 具有两种仿真模式：全系统仿真和系统调用仿真。在全系统仿真模式下，GEM5 可以模拟整个系统，包括所有设备，例如中央处理器、内存以及网卡，还可以运行不加修改的操作系统。在系统调用仿真模式下，GEM5 只可以模拟用户空间的程序。GEM5 可以支持多种指令集，包括 Alpha、ARM、Power、SPARC 以及 64 位 x86 指令集。另外，模拟的 CPU 还可以运行在多个模式，包括单指令单周期、乱序以及按序流水线等模式。为了更好地模拟内存系统，GEM5 将 Ruby 引入其中。Ruby 可以模拟完整的内存系统，并且可以维护内存一致性。例如，Ruby 实现了像 snoopy 和基于目录的 MESI 和 MOESI 等一致性协议。另外，Ruby 还包含 Garnet[1]模块可以用于模拟新型的片上网络互连。Garnet 可以实现典型的网络拓

[1] 一种网络模拟器

扑，例如网状网（mesh）和蝶形网（butterfly），以及常见的路由算法，例如 XY 和 Up-Down 路由等。虽然 GEM5 的功能很强大，但是其在网络方面的支持比较薄弱。例如，虽然 GEM5 可以模拟网卡和链路，但是其对网络拓扑以及路由的自定义则没有支持。

– **Simics**

Simics[9]是一款全系统功能模拟器，它最初由瑞典计算机科学研究所开发，后来该所派生出 Virtutech 公司进行商业化开发，现在 Virtutech 被英特尔收购。Simics 可以模拟多种指令架构，包括 Alpha、x86-64、IA-64、ARM、MIPS (32 位和 64 位)、MSP430、PowerPC (32 位和 64 位)、POWER、SPARC-V8 and V9 和 x86 等；可以模拟众多设备，比如鼠标、键盘、软盘、主板、磁盘、显卡、网卡等。Simics 本身还提供设备模型语言 DML(Device Modeling Language)和相应的编译器 DMLC(Device Modeling Language Compiler)，用户可以根据自己的需求开发自己的虚拟设备。Simics 支持在模拟的硬件上无修改地执行二进制程序，很多操作系统可以在 Simics 模拟的硬件上无修改地执行，包括 MS-DOS、Windows、VxWorks、OSE、Solaris、FreeBSD、Linux、QNX 和 RTEMS 等。Simics 还提供了强大的调试功能，它不仅可以和 gdb、valgrind 这类流行的工具配合，还支持检查点机制，具有反向执行应用程序的能力。Simics 的主要缺点是执行速度慢，而且它是商业产品，不开源，价格昂贵。

– **FeS2**

FeS2[18]模拟器是在 Simics 模拟器的基础上开发的一款时序优先的 X86 指令集多处理器模拟器。该模拟器作为 Simics 的一个模块用于对缓存层次、分支预测以及超标量乱序处理器核的时序进行模拟。FeS2 的特点在于可以对 X86 指令集进行精确模拟，因此对于针对 X86 指令集的应用的研究具有重要意义。另外，FeS2 同样支持全系统模拟，不但可以模拟用户代码，同时可以模拟操作系统指令。这对于研究多线程程序的性能以及调度策略是非常必要的。与 GEM5 相同，FeS2 同样将 Ruby 引入其中。因此，FeS2 也可以进行灵活的内存系统和片上网络系统的研究。与 GEM5 相比，FeS2 的优势在于利用了 Simics 模拟器。因为 Simics 可以保证模拟在功能上的正确性，所以 FeS2 所需要关心的只是系统的时序仿真。事实上，FeS2 实现了 Simics 处理器和内存系统模块的时序模型（timing model）。因此，FeS2 的开发复杂度大幅降低。然而，由于利用了商用非开源系统 Simics，因此会对用户在该模拟器上的二次开发和使用带来一定的困难。

2.2 网络模拟器

网络模拟器是一类主要的局部模拟器，本节详细介绍以下两种最新的网络模拟器。

– **NS3**

NS3[7]是一款开源的离散事件驱动的网络模拟器，主要用于研究和教育领域。NS3 不是 NS2 的扩展，而是一个全新的模拟器，它试图克服 NS2 扩展性差的特点。NS2 使用 C++和脚本语言 OTcl，NS3 使用 C++和脚本语言 Python，并且 NS3 的核心和模拟模块全由 C++编写，Python 仅用于语言绑定和脚本编写。NS3 同时支持 IP 和非 IP 网络，尤其适合无线和 IP 网络模拟，支持 WiFi、WiMax、LTE[2]等网络的物理层和 MAC 层，以及各种静态和动态路由协议如 OLSR 和 AODV。NS3 还支持实时调度，NS3 网络可以与真实网络或主机互连，交换真实网络流量。NS3 强调真实应用和内核代码的重用，不加修改的应用和 Linux 内核网络协议栈可以在 NS3 上运行。此外，NS3 还支持基于 MPI[3]的并行模拟。NS3 是目前最流行

[2] 3GPP Long Term Evolution，第三代移动通信长期演进技术
[3] Message Passing Interface ，一种基于消息传递的并行程序设计标准

网络模拟器之一。

– **OPNET**

OPNET[19]是由 OPNET 科技有限公司开发的商用网络模拟器。该模拟器可以帮助研究人员开发新的网络协议以及验证新协议对系统性能的影响。OPNET 软件包主要包含三个模块：ITDecisionGuru、Modeler 以及 Modeler/radio。其中 ITDecisionGuru 只具有仿真和分析功能，Modeler 在 ITDecisionGuru 基础之上又提供了建库功能，而最全面的 Modeler/radio 则又增加了对移动通信的仿真支持。本质上来说 OPNET 是一个事件驱动的模拟器，可以提供数百种有线和无线协议以及设备模型，例如 X.25、ATM、FDDI、OSPF 等等。除此之外，OPNET 还支持第三方提供的库模块以及用户开发的协议。OPNET 的功能很强大，但是其价格昂贵，并且当网络系统规模比较大时，其仿真速率会大幅下降。

2.3 数据中心模拟器

为了加速云计算和数据中心领域研究的进展，最近有多款模拟器发布，例如美国宾夕法尼亚州立大学的 MDCSim[14]，澳大利亚墨尔本大学和新南威尔士大学以及巴西南里奥格兰德天主教大学联合开发的 CloudSim[15]，卢森堡大学和意大利特兰托大学以及美国北达科他州立大学联合开发的 GreenCloud[16]，以及澳大利亚墨尔本大学开发的 NetworkCloudSim[17]。由于针对云计算不同的问题展开研究，这些模拟器各有所长，下面对他们逐一展开讨论。

– **MDCSim**

MDCSim[14]主要针对多层级数据中心（multi-tier data center）的模拟。在 MDCSim 之前的模拟器具有很多的局限性，例如只能模拟较小规模的系统（扩展性差）或者只能模拟单一的层级。较差的扩展性使得以前的模拟器无法针对大规模的数据中心展开研究。另外，最新的云计算应用大多具有多个层级。例如典型的三层应用模式包含后端、前端和客户端。MDCSim 之前的模拟大多不具有模拟多级数据中心的能力，因此不适用于新型的云计算数据中心。为了解决这两个问题，MDCSim 采用了简单易扩展的三层机构，包括用户层、核心层和通信层。其中，用户层用于模拟应用的多层架构；核心层用于模拟器调度并且用于分析系统的性能，例如吞吐量、延迟、利用率、功耗、可靠性以及热分析等等；通信层主要用于模拟集群内部的通信。虽然 MDCSim 可以模拟多层的应用架构，但是它并不是执行驱动的模拟器，例如它并不模拟真实的应用执行也不模拟诸如中央处理器、内存、交换机以及连线等具体的功能部件。

– **CloudSim**

CloudSim[15]是在原 GridSim[20]模拟器的基础上开发的针对云计算资源分配策略和应用调度算法研究的模拟器。CloudSim 实现了一个层次化的软件架构，包括 SimJava、GridSim、CloudSim 和用户代码四个层次。最底层的 SimJava 是一个离散事件仿真引擎，该引擎为上层提供诸如事件排队和处理服务、系统组件（服务、主机和虚拟机）的创建、组件之间的通信、以及管理模拟时钟等模块。在 SimJava 之上的 GridSim 层主要用于创建高层的软件组件，例如网络、相关的网络性能统计组件、基层的网格组件（包括资源、数据集、负载踪迹以及信息服务）等。在 GridSim 之上的 CloudSim 层主要负责虚拟化的基于云的数据中心环境的建模与仿真，例如虚拟机、内存、存储以及网络带宽的管理端口。一些基础的云计算相关的问题都在这个层级进行模拟，例如虚拟机到物理机的部署、管理应用的执行和系统的动态监控。在软件架构的最顶层是用户代码层，主要负责主机、应用、虚拟机等相关配置和功能，以及中介程序（broker）的调度策略等。但遗憾的是 CloudSim 也不是执行驱动的模拟器，

并不模拟真实的应用，并且同样不支持对类似中央处理器、内存和交换机等功能部件的模拟。

– **GreeenCloud**

根据高德纳（Gartner）的研究报告，数据中心的电能消耗成本占到整个数据中心运行成本的 10%以上[21]，因此最近几年功耗一直是云计算数据中心领域的一个热门的研究话题。当前的大多数研究工作都是针对计算资源消耗的功耗进行研究，而忽略了数据中心网络所消耗的能量。根据 GreeenCloud[16]的作者研究发现，数据中心网络所消耗的能量最大可以占整个数据中心消耗能量的 30%以上。因此，研究低功耗的数据中心网络具有重要的研究意义。然而在 GreenCloud 之前并没有一个针对数据中心网络功耗研究的平台，因此 GreeenCloud 在 NS2[22]模拟器的基础上增加了对服务器、交换机和链路的功耗模拟。然而遗憾的是，GreenCloud 仍然不是执行驱动的模拟器，同样没有对诸如 CPU 和内存等功能部件的细粒度模拟。

– **NetworkCloudSim**

最近开发 CloudSim 的研究人员也逐渐意识到模拟数据中心网络功能的重要性，因此他们在 CloudSim 的基础上增加了网络功能的模拟。NetworkCloudSim[17]的初衷为了克服在其之前的数据中心模拟器存在的一些局限性。例如，MDCSim 和 CloudSim 缺乏对网络的细节模拟，而 CloudSim 和 GreenCloud 则只支持简化的应用模型。为了支持对网络行为的模拟，NetworkCloudSim 在 CloudSim 的基础上增加了交换（switch）模块；同时，为了支持对复杂应用模式的支持，NetworkCloudSim 又增加了 AppCloudlet 模块。然而 NetworkCloudSim 仍然不是执行驱动模拟器，并且对于中央处理器和存储等功能部件的模拟仍然没有涉及。

3 我们设计的模拟框架

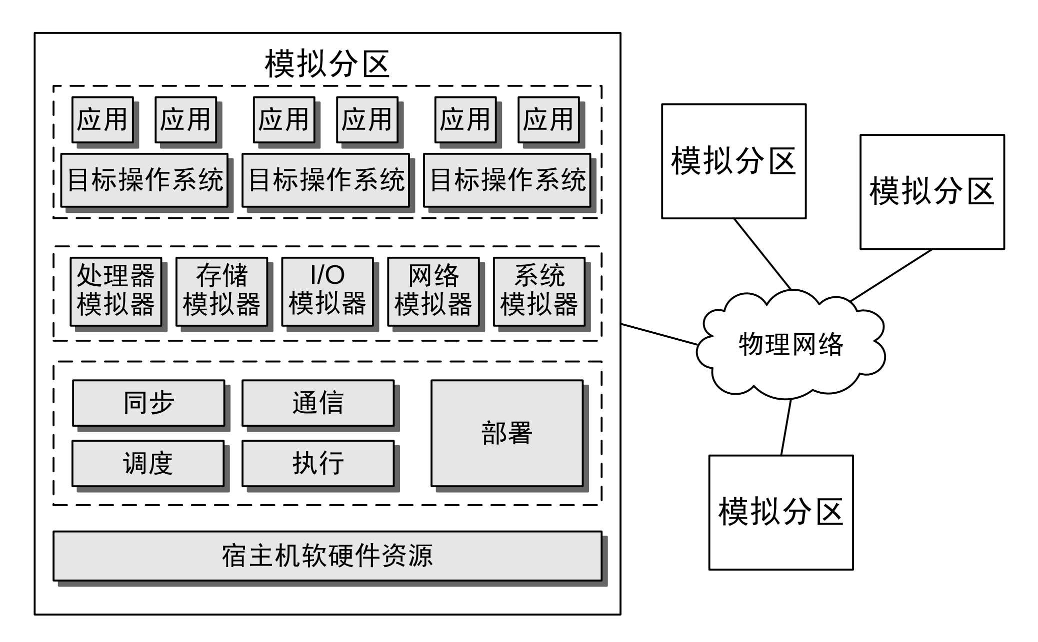


图1. 数据中心全系统模拟框架

为了实现对整个数据中心的模拟，我们提出了一套数据中心全系统模拟框架，利用该框架可以整合现有的各种模拟器，快速地开发出自己的数据中心模拟器。如图 1 所示，该框架是一个分布式结构，每台宿主机上的模拟框架称为一个模拟分区，模拟数据中心的一部分，比如一个机柜（Rack）。不同的模拟分区通过物理网络相连，构建一个完整的数据中心模拟

器。每个模拟分区自下而上分为四层，分别是宿主机软硬件资源、模拟整合层、模拟层和应用层。应用层包括执行在模拟器上的目标系统和应用；模拟层运行各种局部或系统模拟器，为了方便我们称它们为模拟单元。这些模拟单元与外部的接口可能需要少量修改以便调用下一层模拟整合层提供的应用程序接口（API[4]），实现模拟器之间的同步与通信。模拟整合层是整个框架的核心，它包括部署、同步、通信、调度和执行五个模块。该层向上为模拟层中的各模拟单元提供互相同步和通信的服务，向下通过物理网络与其他模拟分区进行同步与通信。下面介绍模拟整合层的各部分。

3.1 部署模块

部署模块的功能是生成目标系统，并将其各部分映射为各模拟单元的任务。部署过程中最重要的问题是如何实现模拟单元划分，这里需要考虑两个因素：1）划分成多少个模拟单元；2）每个模拟单元的规模大小。模拟单元的数目一方面跟目标系统的规模有关，另一方面跟运行模拟器的物理平台资源有关。当模拟单元的数目跟物理资源（总的可用处理器核数）匹配时，模拟器才能获得好的性能。而模拟单元规模大小的划分原则是模拟负载要尽量均匀，这可以防止个别模拟单元的负载太重而成为整个模拟的瓶颈。部署模块具体执行的部署步骤为：1）接收并解析用户对被模拟的目标系统的描述信息，生成目标系统；2）利用划分算法将目标系统划分为组件并映射到模拟单元；3）为互相交互的模拟单元分配通道。

3.2 调度模块

调度模块执行模拟单元的调度。调度的目标是在模拟过程中实现物理平台负载均衡。当总的可用处理器核数目大于等于模拟单元数目时，调度算法将每个模拟单元分配在一个处理器核上执行，以避免多个模拟单元竞争一个处理器核而发生上下文切换的开销。当可用处理器核数小于模拟单元数目时，调度算法让模拟量小的模拟单元共享处理器核，模拟量大的模拟单元独享处理器核。同时，为了减小模拟单元实例间通信的开销，调度算法会将互相通信密切的实例分配在同一台宿主机上。用户可以通过配置文件指定目标系统各部分之间的交互密切程度。

3.3 同步模块

同步模块实现模拟层的模拟单元间以及模拟分区之间的同步。在众多同步的方法中，并行离散事件模拟（Parallel Discrete Event Simulation，PDES）[22]同步方法被广泛使用。在该方法中，每个模拟的逻辑单元称为一个逻辑进程（Logic Process，LP），如果两个逻辑进程之间存在通信则认为它们之间存在一条连接（Link）。其实现同步的基本思想是，如果相连的逻辑进程之间是同步的，则整个系统也是同步的。并行离散事件模拟提供了两种同步机制：保守同步和乐观同步。在保守同步中，每个逻辑进程按照事件时间戳非递减顺序执行事件，每个被执行的事件需要确保是“安全的”，即不会存在比该事件更早的未被执行的事件。而乐观同步则不同，它不保证被执行的事件是安全的，也不保证事件的执行不违反因果关系，而是采用先执行，然后在发现事件的执行违反因果关系时，回滚模拟状态重新执行，以保证行为的正确性。与保守同步相比，乐观同步可以更好地挖掘事件的并行性。因为两个独立的事件，即使不按照时间戳的非递减顺序执行，也不会违反因果关系。在保守同步中这种行为被禁止了，但乐观同步允许这种行为。由于乐观同步需要通过回滚状态来纠正因果关系违例，需要相应的机制来侦测违例的发生，并且需要定时保存模拟状态以允许违例发生时进行回滚和重新执行，因此有可能带来额外开销。

[4] Application Program Interface，

考虑到在大规模模拟的情况下，保存模拟状态需要大量的存储空间，并且侦测违例、保存模拟状态以及状态回滚都需要模拟单元的支持，为了减少对模拟单元的修改，我们采用保守的策略，用基于栅障（Barrier）的同步算法，该算法分为三步：1）计算所有安全的事件的集合；2）按时间戳非递减顺序执行安全事件；3）执行栅障同步操作。每个逻辑进程通过反复执行以上三个步骤实现同步。

3.4 通信模块

通信模块实现模拟层的各模拟单元间以及不同模拟分区之间的数据交换。由于模拟单元间以及模拟分区间的通信都可以看作是进程间的通信，因此传统的实现进程间通信的方法，包括管道、消息队列、共享内存、信号量和套接字等，都可以采用。但考虑到便捷性、可扩展性、通信效率，以及模拟分区间的通信需要跨宿主机，我们提供两种通信方式：FIFO（又称有名管道，即“先入先出”）和套接字。FIFO 主要用于同一宿主机上的两个模拟单元间的通信，而套接字主要用于模拟分区之间的通信。但是我们测试发现套接字的通信效率仅比 FIFO 低 10%左右，因此在存在很多模拟分区的情况下，为了方便，我们常统一采用套接字进行通信。由于频繁的通信会降低整个模拟器的执行效率，为了减少通信次数，在模拟框架中模拟单元或模拟分区间的通信实际仅发生在每次到达同步时，即完成栅障操作后但开始下一回合的同步之前。在上面提到的同步算法中，模拟单元在执行安全事件阶段产生的需要交换的数据会临时放在队列中暂存，在到达同步点后才进行实际的数据交换。为了保证数据交换的及时性，两次同步的间隔应该小于等于数据交换的延迟。

4 评测

为了验证上述模拟框架的有效性，我们基于该框架开发了一个数据中心全系统模拟器。在该模拟器中，一个单节点系统模拟器被用来模拟计算节点的行为，一个网络模拟器被用来模拟互连网络。计算节点和互连网络通过模拟框架连接在一起，构建出用户指定规模的数据中心。我们通过该模拟器模拟一个多节点计算机系统，并与一个相同规模的物理系统进行对比，来验证该模拟器的有效性。受限于我们现有的物理资源，我们仅模拟了一个由一个交换节点连接了四个计算节点的小规模计算机系统，并与一个由单个交换机相连了四个计算节点的物理系统进行对比。由于我们使用的单节点系统模拟器尚不能模拟与现有物理节点配置相同的计算节点，我们目前仅通过执行 Iperf 和 IMB 这两个基准测试程序（Benchmarks）来对这两个系统的通信行为进行对比。

4.1 Iperf 评测

Iperf[24]是一个网络性能评测工具，它采用客户端/服务端模式对点对点网络性能进行测试。在目标系统中，我们随机选择两个计算节点作为 Iperf 的客户端和服务端，通过客户端向服务端发送指定速率的 UDP[5]数据流，并观察实际吞吐率。实验结果如图 2 所示，横坐标是设定的数据流注入速率，纵坐标是实际吞吐率。结果显示，模拟系统和物理系统的实际吞吐率曲线几乎完全一致。

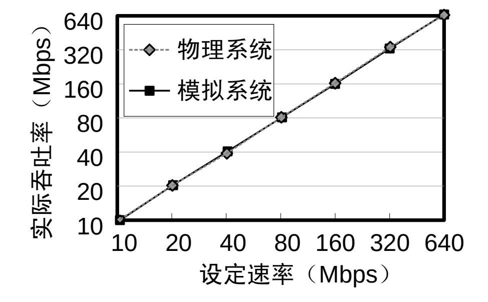


图2. Iperf 测试中实际吞吐率比较

5 User Datagram Protocol,用户数据报协议

4.2 IMB 评测

IMB[25] (Intel® MPI Benchmarks, 英特尔 MPI 性能指示评测)是一套对各种 MPI 通信操作性能进行评测的工具，我们选择了其中的 Pingpong、Bcast 和 All-To-All 三个测试用例对目标系统进行测试，它们分别对应一对一、一对多和多对多三种典型的通信模式。

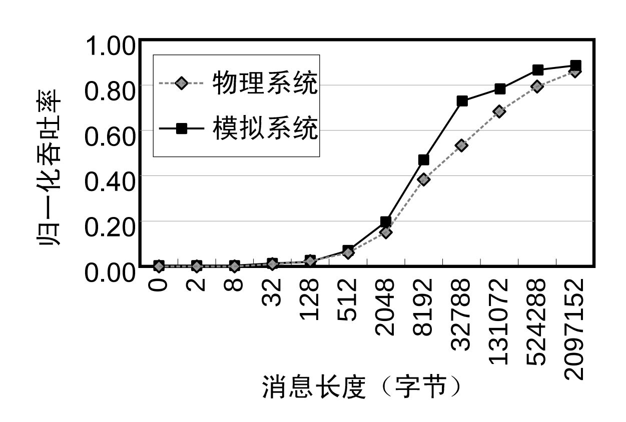


图3. Pingpong 测试中吞吐率随消息长度的变化趋势比较

对于点对点通信，我们在目标系统中随机选择任意一对节点，测试不同消息长度的情况下执行 Pingpong 操作的性能。图 3 显示了 Pingpong 测试中吞吐率随消息长度变化的趋势，其中吞吐率对链路带宽进行了归一化。结果显示模拟系统与物理系统的吞吐率变化曲线仅在局部略有差异，总体表现出了相同的变化趋势。

为测试一对多的通信性能，我们在目标系统中依次让各节点作为根节点，向其他所有节点发送指定长度的消息进行 Bcast 测试。用完成一次 Bcast 操作的时间作为 Bcast 的性能指标。为了便于比较，将结果对最长 Bcast 操作时间进行了归一化。图 5 显示了在物理系统和模拟系统中执行 Bcast 时，Bcast 完成时间随消息长度变化的曲线，两者表现出同样的变化趋势。

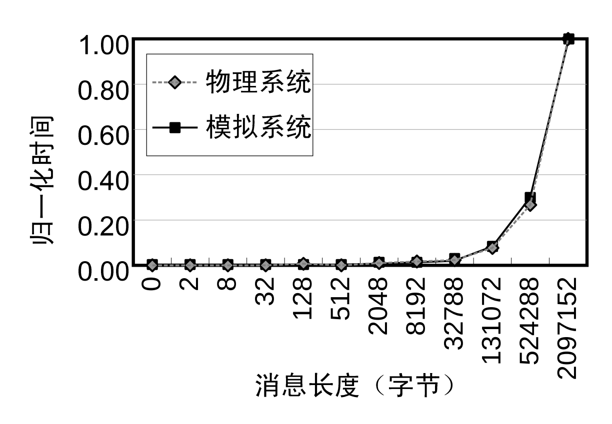


图4. Bcast 操作时间随消息长度的变化趋势比较

在测试多对多的通信性能时，在目标系统中，每个节点向其他所有节点发送指定长度的消息。所有节点间完成一轮通信的时间被作为 All-To-All 操作的性能。同样，我们将结果对最长 All-To-All 操作时间进行归一化。图 5 同样表明模拟系统的行为与物理系统高度一致。

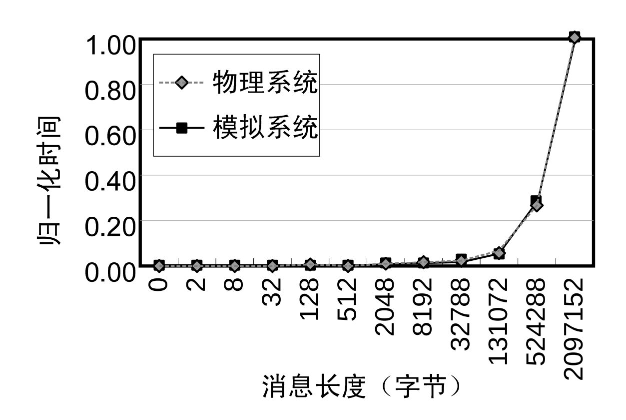


图5. All-To-All 操作时间随消息长度的变化趋势比较

由上述的实验可知，基于我们提出的数据中心模拟框架开发的数据中心模拟器模拟物理系统，其结果与物理系统行为高度一致。因此基于该框架开发数据中心模拟器是切实可行的。

5 结论

本文在重用现有模拟器构建数据中心模拟器的基础上构建了一套数据中心全系统模拟

框架，该框架可以整合现有局部模拟器或单节点系统模拟器，构建完整的数据中心模拟器。我们基于该框架开发了一个数据中心模拟器，相关实验显示该模拟器所模拟的计算机系统行为与目标物理计算机系统高度一致。

参考文献：

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the Clouds: A Berkeley View of Cloud Computing, Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (February 2009)

[2] K. Kant, “Data Center Evolution: A Tutorial on State of the Art, Issues, and Challenges”, Computer Networks Journal, Vol 53, No 17, Dec 2009.

[3] M-Sim: The Multithreaded Simulator. http://www.cs.binghamton.edu/~jsharke/m-sim/

[4] Dinero IV, Trace-Driven Uniprocessor Cache Simulator. http://pages.cs.wisc.edu/~markhill/DineroIV/

[5] Milo, M.K.M., et al., Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. SIGARCH Comput. Archit. News, 2005. 33(4): p. 92-99

[6] DiskSim：http://www.pdl.cmu.edu/DiskSim/

[7] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley. ns-3 project goals. In WNS2 ’06: Proceeding from the 2006 workshop on ns-2: the IP network simulator, page 13, New York, NY, USA,2006. ACM.

[8] Brooks, D., V. Tiwari, M. Martonosi, Wattch: a framework for architectural-level power analysis and optimizations. ISCA, 2000: p. 83-94

[9] Magnusson, P.S.C., M.; Eskilson, J.; Forsgren, D. Simics: A full system simulation platform. Computer IEEE, 2002. vol.35, no.2: p. 50-58

[10] Todd Austin, E.L., Dan Ernst. SimpleScalar: An Infrastructure for Computer System Modeling. Computer IEEE, 2002. vol. 35, no. 2: p. 59-67

[11] Stephen, A.H. Using Complete Machine Simulation to Understand Computer System Behavior. 1998, Stanford University

[12] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, et al. The M5 Simulator: Modeling Networked Systems, IEEE Micro, 2006, P.: 52-60

[13] N. Binkert, B. Beckmann, G. Black, et al. The gem5 Simulator, ACM SIGARCH Computer Architecture News archive, Volume 39 Issue 2, May 2011,Pages 1-7

[14] S.H. Lim, B. Sharma, G. Nam, E. Kim, and C.R. Das, “MDCSim: A Multi-Tier Data Center Simulation Platform”, In Proceeding of IEEE International Conference on Cluster Computing and Workshops, pp: 1-9, 2009.

[15] CloudSim: R. Buyya, R. Ranjan, and R.N. Calheiros, “Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities”, In Proceeding of International Conference on High Performance Computing & Simulation, pp: 1-11, 2009.

[16] D. Kliazovich, P. Bouvry, Y. Audzevich, and S.U. Khan, “GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers”, In Proceeding of IEEE Global Telecommunications Conference, pp: 1-5, 2010.

[17] S.K. Garg and R. Buyya, “NetworkCloudSim: Modeling Parallel Applications in Cloud Simulations”, In Proceeding of IEEE International Conference on Utility and Cloud Computing, pp: 105-113, 2011.

[18] N. Neelakantam, C. Blundell, J. Devietti, M. M. K. Martin, and C. Zilles. The FeS2 simulator. In Poster session at ASPLOS '08, 2008.

（下转第 16 页）